Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 8306732, 13 pages
https://doi.org/10.1155/2017/8306732

Research Article

Application of Heuristic and Metaheuristic Algorithms in Solving Constrained Weber Problem with Feasible Region Bounded by Arcs

Igor Stojanović,[1] Ivona Brajević,[2] Predrag S. Stanimirović,[2] Lev A. Kazakovtsev,[3] and Zoran Zdravev[1]

[1]*Faculty of Computer Science, Goce Delčev University, Goce Delčev 89, 2000 Štip, Macedonia*
[2]*Department of Mathematics and Informatics, Faculty of Science and Mathematics, University of Niš, Višegradska 33, 18000 Niš, Serbia*
[3]*Department of Systems Analysis and Operations Research, Reshetnev University, Prosp. Krasnoyarskiy Rabochiy 31, Krasnoyarsk 660037, Russia*

Correspondence should be addressed to Predrag S. Stanimirović; pecko@pmf.ni.ac.rs

Received 26 February 2017; Accepted 15 May 2017; Published 14 June 2017

Academic Editor: Domenico Quagliarella

The continuous planar facility location problem with the connected region of feasible solutions bounded by arcs is a particular case of the constrained Weber problem. This problem is a continuous optimization problem which has a nonconvex feasible set of constraints. This paper suggests appropriate modifications of four metaheuristic algorithms which are defined with the aim of solving this type of nonconvex optimization problems. Also, a comparison of these algorithms to each other as well as to the heuristic algorithm is presented. The artificial bee colony algorithm, firefly algorithm, and their recently proposed improved versions for constrained optimization are appropriately modified and applied to the case study. The heuristic algorithm based on modified Weiszfeld procedure is also implemented for the purpose of comparison with the metaheuristic approaches. Obtained numerical results show that metaheuristic algorithms can be successfully applied to solve the instances of this problem of up to 500 constraints. Among these four algorithms, the improved version of artificial bee algorithm is the most efficient with respect to the quality of the solution, robustness, and the computational efficiency.

1. Introduction

The Weber problem is one of the most studied problems in location theory [1–3]. This optimization problem searches for an optimal facility location $X^* \in \mathbb{R}^2$ on a plane, which satisfies

$$X^* = \arg\min_{X\in\mathbb{R}^2} f(X) = \arg\min_{X\in\mathbb{R}^2} \sum_{i=1}^{N} w_i \|A_i - X\|. \quad (1)$$

In (1), it is assumed that $A_i \in \mathbb{R}^2$, $i \in \{1, \ldots, N\}$ are known demand points, $w_i \in \mathbb{R}$ and $w_i \geq 0$ are weight coefficients, and $\|\cdot\|$ is a matrix norm, used as the distance function.

The basic Weber problem is stated with the Euclidean norm underlying the definition of the distance function. Also, many other types of distances have been used in the facility location problems [3–5]. In general, a lot of extensions and modifications of the Weber location problem are known. Detailed reviews of these problems can be found in [3, 6].

The most popular method for solving the Weber problem with Euclidean distances is given by a one-point iterative procedure which was first proposed by Weiszfeld [7]. Later, Vardi and Zhang developed a different extension of Weiszfeld's algorithm [8], while Szegedy partially extended Weiszfeld's algorithm to a more general problem [9]. In particular, some variants of the continuous Weber problem represent nonconvex optimization problems which are hard to be solved exactly [10]. A nonconvex optimization problem may have multiple feasible regions and multiple locally optimal points

within each region [11]. Consequently, finding the global solution of a nonconvex optimization problem is very difficult.

Heuristics and metaheuristics represent the main types of stochastic methods [12]. Both types of algorithms can be used to speed up the process of finding a high-quality solution in the cases where finding an optimal solution is very hard. The distinctions between heuristic and metaheuristic methods are inappreciable [12]. Heuristics are algorithms developed to solve a specific problem without the possibility of generalization or application to other similar problems [13]. On the other hand, a metaheuristic method represents a higher-level heuristic in the sense that they guide their design. In such a way we can use any of these methods to design a specific method for computing an approximate solution for an optimization problem.

In the last several decades, there is a trend in the scientific community to solve complex optimization problems by using metaheuristic optimization algorithms. Some applications of metaheuristic algorithms include neural networks, data mining, industrial, mechanical, electrical, and software engineering, as well as certain problems from location theory [14–21]. The most interesting and most widely used metaheuristic algorithms are swarm-intelligence algorithms which are based on a collective intelligence of colonies of ants, termites, bees, flock of birds, and so forth [22]. The reason of their success lies in the fact that they use commonly shared information among multiple agents, so that self-organization, coevolution, and learning during cycles may help in creating the highest quality results. Although not all of the swarm-intelligence algorithms are successful, a few techniques have proved to be very efficient and thus have become prominent tools for solving real-world problems [23]. Some of the most efficient and the most widely studied examples are ant colony optimization (ACO) [24–26], particle swarm optimization (PSO) [15, 27–29], artificial bee colony (ABC) [19, 30–35], and recently proposed firefly algorithm (FA) [18, 36–38] and cuckoo search (CS) [17, 39–41].

Different heuristic methods are proposed in order to provide encouraging results for challenging continuous Weber problem with regard to solution quality and computational effort [42–46]. Also, some variants of the Weber problem have been successfully solved by different metaheuristic approaches [47–52]. In [52], the authors studied a capacitated multisource Weber problem as an extended facility location problem that involves both facility locations and service allocations simultaneously. The method proposed in [52] is based on the integration of two genetic algorithms. The problem of locating one new facility with respect to a given set of existing facilities in the plane and in the presence of convex polyhedral barriers was considered in [47]. The general strategy in [47] arises from the iterative application of a genetic algorithm for the subproblems selection. A hybrid particle swarm optimization approach was applied in solving the incapacitated continuous location-allocation problem in [48]. In [49], the authors compared performances of four metaheuristic algorithms, modified to solve the single-facility location problem with barriers. The method for solving a kind of Weber problem from [50] was developed using an evolutionary algorithm enhanced with variable neighborhood search.

The aim of this paper is to investigate the performances of some prominent swarm-intelligence metaheuristic approaches to solve the constrained Weber problem with feasible region bounded by arcs. This variant of Weber problem has a nonconvex feasible set given by the constraints that make it much harder to find the global optimum using any deterministic algorithms. Hence, metaheuristic optimization algorithms can be employed in order to provide promising results.

In this paper, four swarm-intelligence techniques are applied to solve this version of the constrained Weber problem: the artificial bee colony for constrained optimization [53], the crossover-based artificial bee colony (CB-ABC) algorithm [54], the firefly algorithm for constrained optimization [37], and the enhanced firefly algorithm (E-FA) [55]. The CB-ABC and the E-FA are two of the most recently proposed improved variants of the ABC and FA for solving constrained problems, respectively. Also, a heuristic algorithm is proposed in [44] with the aim of solving this version of the constrained Weber problem. Hence, it is also implemented for the purpose of comparison with the metaheuristic approaches. These five techniques are tested to solve randomly generated test instances of constrained Weber problem with feasible region bounded by arcs of up to 500 constraints.

The rest of the paper is organized as follows. A formulation of the constrained Weber problem with feasible region bounded by arcs and the heuristic approach developed to solve this variant of the constrained Weber problem are presented in Section 2. Section 3 presents the four metaheuristic optimization techniques used to solve this variant of the Weber problem. Description of the generated benchmark functions and comparative results of the four implemented metaheuristic techniques are given in Section 4. Concluding remarks are provided in Section 5.

2. The Heuristic Method for Solving a Constrained Weber Problem

The constrained Weber problem with feasible region bounded by arcs in the continuous space was introduced in [44]. In order to complete our presentation, we briefly restate the method. It can be formulated by the goal function defined in (1) and by the feasible region which is defined on the basis of constraints of two opposite types:

$$\begin{aligned} \mathcal{S}_{<} &= \left\{ i \in \{1, \ldots, N\} \mid \left\| X - A_i \right\| \leq 1 \right\}, \\ \mathcal{S}_{>} &= \left\{ i \in \{1, \ldots, N\} \mid \left\| X - A_i \right\| \geq 1 \right\}, \end{aligned} \tag{2}$$

where N is the total number of demand points and $\{1, \ldots, N\}\mathcal{S}_{<}$ and $\mathcal{S}_{>}$ are subsets of the set of demand point indices satisfying $\{1, \ldots, N\}\mathcal{S}_{<}$, $\mathcal{S}_{>} \subseteq \{1, \ldots, N\}$, and $\mathcal{S}_{<} \cap \mathcal{S}_{>} = \emptyset$. For the sake of simplicity, the optimization problem given by (1) with constraints (2) is denoted as the CWP problem.

Such a problem may occur if some demand points coincide with locations of some important facilities and the

searched optimal location X^* must be close to them. Other demand points may coincide with dangerous facilities and the facility X^* must be located far from them.

The metric used in practically important location problems depends on various factors, including properties of the transportation means [44]. In the case of public transportation systems, the price usually depends on a distance. However, some minimum price is usually defined. For example, the initial fare of the taxi cab may include some distance, usually 1–5 km. Having rescaled the distances so that this distance included in the initial price is equal to 1, we can define the price function d_P as

$$d_P(X,Y) = \max\{\|X - Y\|, 1\} \quad \forall X, Y \in \mathbb{R}^2, \tag{3}$$

where $\|\cdot\|$ is a matrix norm.

In the case of distance function defined by (3), the problem can be decomposed into series of constrained location problems with the Euclidean metric where the area of the feasible solutions is bounded by arcs. Each of the problems has the feasible region equal to the same intersection of the discs with centers in the demand points. For more details, see [44, 56].

The Weiszfeld procedure for solving the Weber problem with a given tolerance ε, based on the results from [57], is presented as Algorithm 2.1 in [44].

An algorithm based on the Weiszfeld procedure for solving the CWP defined by objective (1) and constraints (2) was proposed [44]. The feasible set of our constrained optimization problems is generally nonconvex, while the objective function $f(X)$ given by (1) is convex [58]. A solution of constrained optimization problems with convex objective functions coincides with the solution of the unconstrained problem or lies on the border of the forbidden region [59]. Thus, if X^* is a solution of the constrained problem given by (1) with constraints (2) then it is the solution of the unconstrained problem (1) or $\exists i \in \{\overline{1,N} : \|A_i - X^*\|_2 = 1\}$.

Step 2.2 of Algorithm 2.1 from [44] can lead to generating a new point X^{**} outside the feasible region determined by constraints (2). Let us denote this region $\mathscr{R}_f$. It is assumed that $\mathscr{R}_f \neq \emptyset$.

For an arbitrary point $X \in \mathbb{R}^2$, let us denote the closest point in $\mathscr{R}_f$ by $\mathscr{C}(X)$. It can be computed using

$$\mathscr{C}(X) = \arg\min_{X' \in \mathscr{R}_f} \left\|X - X'\right\| = \begin{cases} X, & X \in \mathscr{R}_f, \\ \arg\min_{X' \in \mathscr{R}_f} \left\|X - X'\right\|, & X \notin \mathscr{R}_f. \end{cases} \tag{4}$$

Algorithm 1 was proposed as Algorithm 2.2 in [44], and it is based on the substitution of the point X^{**} generated in Step 2.2 of Algorithm 2.1 from [44] with its closest point $\mathscr{C}(X^{**})$ in the feasible region.

3. Review of the Metaheuristic Optimization Techniques

The four metaheuristics used to solve constrained Weber problem with feasible region bounded by arcs are described in the following subsections.

3.1. Artificial Bee Colony Algorithm for Solving the CWP. A numerical variant of the ABC algorithm for constrained optimization problems (COPs) proposed in [60] is applied to solve the CWP. In the ABC the population is iteratively refined through employed, onlooker, and scout bee phases.

The update process used in the employed and onlooker bee phase is the same and it is determined by

$$v_{ij} = \begin{cases} x_{ij} + \varphi_j \cdot \left(x_{ij} - x_{kj}\right), & \text{if } R_j < \text{MR} \\ x_{ij}, & \text{otherwise,} \end{cases} \tag{5}$$

where φ_j is a uniform random number in the range $[-1, 1]$, x_k represents another solution selected randomly from the population, MR is the modification rate control parameter, R_j is a randomly chosen real number in the range $[0, 1)$, and $j = 1, 2$. The update process is completed when the selection between x_i and v_i is carried out.

The ABC uses Deb's rules in order to decide which solution will be kept for the next iteration. This constraint handling method consists of a set of three feasibility rules introduced by Deb [61]. They are the following: (1) any feasible solution is preferred to any infeasible solution, (2) between two feasible solutions, the one having a better fitness value is preferred, and (3) if both solutions are infeasible, the one with the lowest sum of constraint violations is preferred.

In the employed bee phase, every solution involves the update process. On the other hand, in the onlooker bee phase only the solutions selected probabilistically proportional to their fitness values have the chance to be upgraded [60].

In the scout phase solutions that do not improve over a certain number of cases are replaced by new randomly generated solutions. The control parameters *limit* and the scout production period SPP are used in this phase. The parameter *limit* is used to signify exhausted food source, while SPP parameter is employed in order to denote a predetermined period of cycles for producing scout bees.

The pseudocode of the ABC is given as Algorithm 2.

3.2. Crossover-Based Artificial Bee Colony Algorithm for Solving the CWP. Recent improved variant of the ABC for COPs, called crossover-based artificial bee colony, is also used to solve the constrained Weber problem [54]. The main modifications introduced in the CB-ABC are related to the search operators used in each bee phase in order to improve the distribution of good information between solutions [54]. The differences between the CB-ABC and the ABC for COPs are given as follows.

In the employed bee phase, the CB-ABC algorithm uses modified search equation (5), in which φ is the same random number of each parameter j which will be changed. Also, the CB-ABC does not use the fixed value of MR control parameter. Value of MR linearly increases from 0.1 to the predefined

Require: Coordinates and weights of the demand points $A_i = (a_1^i, a_2^i), w_i,\ i = \overline{1, N}$, pre-specified tolerance ε, constraints (2) specified by sets $\mathcal{S}_<$ and $\mathcal{S}_>$.

Step 1. Calculate the initial point $X^* \in \mathcal{R}_f$ (here, $\mathcal{R}_f$ is the feasible set bounded by constraints);

$$X^* = \mathcal{C}(X^*);\ \Delta = +\infty.$$

Step 2. While $\Delta > \varepsilon$ do:

Step 2.1. $n_{\text{iter}} = n_{\text{iter}} + 1$;

$$d_{\text{denom}} = \sum_{i=1}^{N} \frac{w_i}{\|A_i - X^*\|_2}.$$

Step 2.2. $x_r^{**} = \sum_{i=1}^{N} (x_r^* w_i / (\|X^* - A_i\|_2 \cdot d_{\text{denom}}))\ \forall r \in \{1, 2\}$.

Step 2.3. If $X^{**} \notin \mathcal{R}_f$ then $X^{**} = \mathcal{C}(X^{**})$.

Step 2.4. $\Delta = \|X^* - X^{**}\|$; $X^* = X^{**}$.

Step 2.5. Continue Step 2.

Step 3. STOP, return X^{**}.

Algorithm 1: Solving the CWP problem.

Initial parameters of the ABC including maximum cycle number (MCN), SN, MR, *limit*, SPP;
Generate initial population x_i ($i = 1, 2, \ldots, \text{SN}$) randomly in the search space and evaluate each x_i;
$t = 0$;
while ($t <$ MCN) **do**
  for $i = 1$ to SN **do**
    Generate a solution v_i with x_i by Eq. (5), evaluate it and apply selection process based on Deb's method between v_i with x_i;
  end for
  for $i = 1$ to SN **do**
    Select food source x_l based on fitness proportionate selection;
    Generate a solution v_l with x_l by Eq. (5), evaluate it and perform selection process based on Deb's method between v_l with x_l;
  end for
  if (t mod SPP = 0) **then**
    Every solution which did not enhance at least *limit* number of times is replaced, each with a randomly produced solution.
  end if
  Memorize the best solution reached so far.
  $t = t + 1$
end while

Algorithm 2: Pseudocode of the ABC.

Table 1: The values of specific control parameters of the algorithms.

FA		E-FA		ABC		CB-ABC	
α	0.25	α	0.25	MR	0.8	$\text{MR}_{\max}$	0.9
β	1	β	1.5	*limit*	SN	*limit*	1
γ	1	γ	1	SPP	SN	SPP	50
						P	0.3

value $\text{MR}_{\max}$ in the first $P * \text{MCN}$ iterations, while the value $\text{MR} = \text{MR}_{\max}$ is used in the remaining iterations. The value of P is defined in Table 1.

In the onlooker bee phase, the CB-ABC proposes a new search equation with the aim of enabling better exploration of the neighborhood of the high-quality solution. This equation is given by

$$v_{ij} = x_{ij} + \varphi \cdot \left(x_{lj} - x_{kj}\right), \tag{6}$$

where φ is a uniform random number in range $[-1, 1]$, x_l and x_k represent the other two solutions selected randomly from the population, R_j is a randomly chosen real number in the range $[0, 1)$, and $j = 1, 2$.

In the scout bee phase, the CB-ABC uses uniform crossover operator to generate new solutions in a promising region of the search space. Therefore, after each SPPth iteration, each solution x_i which did not improve *limit* number of times is replaced with a new solution which is created by

$$v_{ij} = \begin{cases} y_j, & \text{if } R_j < 0.5 \\ x_{ij}, & \text{otherwise,} \end{cases} \tag{7}$$

where y_j is the jth element of the global best solution found so far, R_j is a randomly chosen real number in range $[0, 1)$, and $j = 1, 2$.

Initial parameters of the FA including SN, MCN, $\alpha_0, \beta_0, \gamma$.
Generate initial population of fireflies x_i ($i = 1, 2, \ldots, \text{SN}$) randomly distributed in the solution space.
Assume that $\varphi(x_i)$ is the expanded objective function of x_i ($i = 1, 2, \ldots, \text{SN}$) calculated by (10).
$t = 0$.
while $t <$ MCN **do**
 for $i = 1$ to SN **do**
  for $j = 1$ to SN **do**
   if $(\varphi(x_j) < \varphi(x_i))$ **then**
    Generate a new x_i according to Eq. (8) and evaluate it.
   end if
  end for
 end for
 $t = t + 1$.
 $\alpha(t) = \alpha(t-1) \cdot (10.0^{-4.0}/0.9^{1/\text{MCN}})$.
 Rank the fireflies and memorize the best solution achieved so far.
end while

Algorithm 3: Pseudocode of the FA.

3.3. Firefly Algorithm for Solving the CWP. In order to solve the CWP we have employed a numerical optimization version of the FA for COPs, introduced in [37]. In the FA, a colony of artificial fireflies searches for good solutions in every iteration.

The search operator represents the movement of a firefly i to another more attractive or brighter firefly j and it is given by

$$x_{ik} = x_{ik} + \beta \cdot \left(x_{jk} - x_{ik}\right) + \alpha \cdot S_k \cdot \left(\text{rand}_k - \frac{1}{2}\right), \quad (8)$$

where the second term is due to the attraction and the third term is a randomization term.

In the second term of (8), the parameter β is the attractiveness of fireflies which is calculated according to the following monotonically decreasing function [62]:

$$\beta = \beta_0 \cdot e^{-\gamma \cdot r_{ij}^2}, \quad (9)$$

where r_{ij} denotes the distance between firefly x_i and firefly x_j, while β_0 and γ are predetermined algorithm parameters: maximum attractiveness value and absorption coefficient, respectively. Distance between fireflies is calculated by the Euclidean distance.

In the third term of (8), $\alpha \in [0, 1]$ is a randomization parameter, S_k are the scaling parameters, and rand_k is a random number uniformly distributed between 0 and 1. The scaling parameters S_k ($k = 1, 2$) are calculated by $S_k = |u_k - l_k|$, where l_k and u_k are the lower and upper bound of the parameter x_{ik}. Diversity of solutions is controlled by the randomization parameter α which needs to be reduced gradually during iterations so that it can vary with the iteration counter t [63].

In the FA for solving CWP, penalty functions approach is used in order to handle the constraints. In this way, a constrained problem is solved as an unconstrained one. A general formula of calculation penalty functions is given in [64] by

$$\varphi(X) = f(X) + \sum_{j=1}^{q} r_j \cdot \max\left(0, g_j(X)\right)^2 + \sum_{j=q+1}^{m} c_j \cdot \left|h_j(X)\right|, \quad (10)$$

where $\varphi(X)$ is the new (expanded) objective function to be optimized, r_j and c_j are positive constants normally called "penalty factors," q is the number of inequality constraints, and $m - q$ is the number of equality constraints for a given problem. We found it suitable to set each r_i to the value $r_i = 10^8$. The penalty factors for equality constraints were not used, since these problems have only inequality constraints.

The pseudocode of the FA is given as Algorithm 3.

3.4. An Enhanced Firefly Algorithm for Solving the CWP. An enhanced firefly algorithm for COPs is presented in [55] and it is also applied to solve the CWP. Two modifications are incorporated in the E-FA in order to improve the performance of the firefly algorithm for COPs.

The first modification is related to using Deb's rules instead of the penalty approach. Three feasibility rules are employed instead of the greedy selection in order to decide which firefly is brighter. These rules are also used each time after (8) is applied in order to decide whether the solution will be updated. Evaluation of solution population is given as Algorithm 4.

The second modification is employing the geometric progression reduction scheme to reduce the scaling factors S_k at the end of each cycle, by the rule

$$S_k(t) = S_k(t-1) \cdot \theta^{1/\text{MCN}}, \quad (11)$$

where MCN is the maximum cycle number, t is the current iteration number, and $\theta = 10.0^{-4.0}/0.9$.

```
for i = 1 to SN do
  for j = 1 to SN do
    if (x_j is better than x_i based on Deb's rules) then
      for k = 1 to D (dimension of the problem) do
        g_k = x_ik + β · (x_jk − x_ik) + α · S_k · (rand_k − 1/2)
      end for
      if (g is better than x_i based on Deb's rules) then
        x_i = g {the solution is updated}
      end if
    end if
  end for
end for
```

Algorithm 4: Evaluation of a new population in the E-FA.

4. Experimental Study

The ABC, CB-ABC, FA, and E-FA are implemented in the Java programming language on a PC Intel Core i5-3300@3 GHz with 4 GB of RAM. The heuristic algorithm based on the modified Weiszfeld procedure is also implemented for the purpose of comparison with the metaheuristic approaches.

4.1. Benchmark Functions. The performance of the four metaheuristics techniques and behavior of the heuristic algorithm are evaluated through eighteen test instances of the single-facility constrained Weber problems with the connected feasible region bounded by arcs with equal radius.

The benchmark problems with the increasing number of input points are randomly generated according to the algorithm given in [44]. These problems have 5, 10, 50, 100, 250, and 500 input points. Three different random test problems are generated for each number of input points. Hence, these test instances have a nonconvex feasible set given from 5 up to 500 constraints.

Four example problems, named P1, P4, P7, and P10 with 5, 10, 50, and 100 input points, respectively, are shown in Figure 1. In each test image, the feasible region is represented by a gray surface area and the final solution obtained by the heuristic algorithm [44] is represented by a red cross.

4.2. Parameter Settings. The solution number (SN) in the four metaheuristic algorithms was set to 20. The maximum number of fitness function evaluations (FEs) was used as the stopping criterion. The allowed FEs were set to 8000. In addition, the metaheuristic algorithms presented in Section 2 have several other control parameters that considerably influence their performance. The values of these control parameters are presented in Table 1.

In order to calculate FEs researchers usually use the rule SN∗MCN, where MCN is the maximum number of iterations [65, 66]. Hence, the FA and E-FA were terminated after 400 iterations. The number of consumed fitness evaluations in each iteration of the ABC and CB-ABC algorithms is 2 ∗ SN, since it calculates the solutions both in the employed bee and in onlooker bee phase [65]. Therefore, to ensure a fair comparison, the ABC and CB-ABC algorithms were terminated after 200 iterations.

For the FA, it is widely reported in the literature that the light absorption coefficient $\gamma = O(1)$, the initial attractiveness $\beta_0 = 1$, and the initial randomness factor $\alpha_0 \in [0, 1]$ can be used for most applications [36, 62]. It can be seen from Table 1 that the value of the parameter γ was set to 1 and the initial value of α was set to 0.25 for both FA and E-FA. A typical value of $\beta_0 = 1$ is used in the FA. It was empirically determined that slightly higher value of the parameter β_0 is more suitable for the E-FA. Hence $\beta_0 = 1.5$ was adapted. For the ABC and CB-ABC algorithms, the values of the specific control parameters were taken from [53, 54], where these algorithms were proposed to solve COPs. Especially for the CB-ABC, it was empirically determined that a lower value of the scout production period SPP is more appropriate for solving the CWP. Therefore, it was set to 50. Each of the experiments was repeated for 30 runs.

4.3. Analysis of Solution Quality and Robustness. The coordinates of the solution, corresponding objective function value, and the CPU time (in seconds) obtained by the heuristic algorithm are arranged in Table 2. To analyze the solution quality of the tested four metaheuristic algorithms, the best values, mean values, and standard deviations have been obtained by the ABC, CB-ABC, FA, and E-FA algorithms over 30 runs. Significance tests are used to achieve reliable comparisons. According to [67], two-sample 95%-confidence t-test was conducted between each pair of compared metaheuristics on every benchmark function. The calculated best results are presented in Table 3, while the mean values and standard deviations are arranged in Table 4. Results of two-sample t-tests are reported in Table 5. The sign "+" indicates that the associated comparative algorithm is significantly better than the other one, while the sign "−" indicates it is significantly worse than the opposite one. If both algorithms show similar performance, they are both marked by "+."

Kazakovtsev in [44] experimentally proved the convergence of the heuristic algorithm on randomly generated test problems. Hence, the calculated best values of the metaheuristics can be compared to the results found by the heuristic approach in order to show the ability of the metaheuristic algorithm to reach the near-optimal result. The obtained mean and standard deviation values indicate the robustness of the metaheuristic approaches.

It can be seen from Table 3 that each of the metaheuristic algorithms found the best results which are very close to the results obtained by the heuristic algorithm. More precisely, the FA obtained 10 better best results (P2, P4, P5, P6, P8, P9, P11, P16, P17, and P18) and 8 worse best results with respect to the heuristic approach. The E-FA obtained 11 better best results (P1, P2, P4, P5, P6, P8, P9, P11, P16, P17, and P18), one equal best result (P3) and 6 slightly worse best results in comparison with the heuristic approach. The ABC algorithm achieved 12 better best results (P2, P4, P5, P6, P8, P9, P10, P11, P12, P16, P17, and P18), one equal best result (P3), and 5 worse best results with respect to the heuristic approach. The algorithm CB-ABC was able to find better or the same best solution for all problems with respect to the heuristic

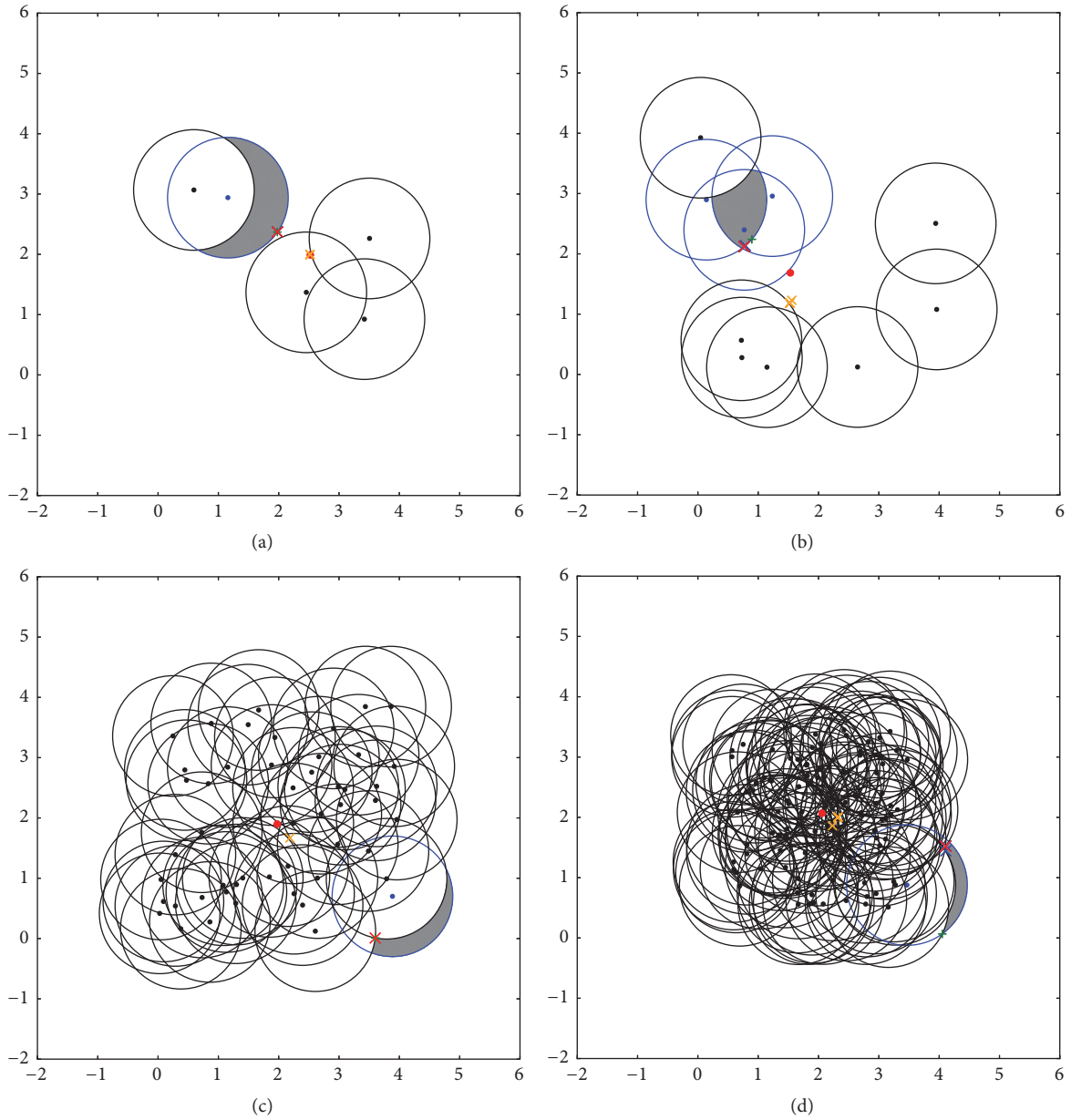

Figure 1: Example problems with (a) 5 points (P1), (b) 10 points (P4), (c) 50 points (P7), and (d) 100 points (P10).

algorithm, with the exception of the problem P7, where the CB-ABC obtained slightly worse best result.

In terms of best results from Table 3, it can be noticed that the CB-ABC achieved better and in several cases the same values in comparison with each considered metaheuristic approach. Further, each of the improved metaheuristics, the E-FA and CB-ABC, obtained better best results with respect to both original metaheuristic algorithms for the majority of test problems. If we compare the performance of the original ABC to that of the original FA, it can be seen that both algorithms show similar ability to reach the near-optimal result; that is, the ABC has found 9 slightly better best results and 9 slightly worse ones compared to the FA.

From Table 4, it can be seen that mean and standard deviation results obtained by the CB-ABC are much better than the results obtained by the other metaheuristic algorithms. The CB-ABC converged consistently to the same solution with the same objective function value and very lower standard deviation. If we compare the robustness of the remaining three metaheuristics, it can be noticed that the E-FA outperformed the FA and ABC. Compared with the ABC, the FA obtained 9 better mean results and standard deviation values (P1, P2, P4, P6, P11, P13, P14, P17, and P18). The remaining mean and standard deviation results are better in the case of the ABC algorithm, with the exception of P5 and P15 where the FA and ABC show similar performances.

Table 2: The solution point, corresponding objective function, and time in seconds provided by the heuristic algorithm.

Prob	D	Solution point	Objective function value	Time (sec)
P1	5	{1.97900484015, 2.37038768628}	38.6308975914	$4.61E-05$
P2	5	{2.511104376, 2.36822367838}	35.6524439425	$3.34E-05$
P3	5	{3.29440267812, 1.91712629136}	32.8210021279	$3.45E-05$
P4	10	{0.768148485664, 2.11871223533}	18.9069002076	0.0001
P5	10	{1.35331974755, 1.96480428579}	18.4514161651	0.0001
P6	10	{2.40607643306, 1.99679256415}	106.1627295736	0.0001
P7	50	{3.60036240333, 0.00927504530255}	723.0047301353	0.0002
P8	50	{2.14432492473, 0.0678576078541}	114.6249767471	0.0007
P9	50	{1.61875017517, 0.463308206334}	455.1344852622	0.0002
P10	100	{4.10439880225, 1.51968295711}	231.0855570171	0.0003
P11	100	{2.85555816286, 4.68224060226}	297.6045429275	0.0009
P12	100	{2.81772843636, 4.5680021125}	286.7386225628	0.0005
P13	250	{0.861049499813, 0.377164213194}	549.9465893405	0.0008
P14	250	{0.794900743781, 0.459653174322}	543.7610648263	0.0007
P15	250	{0.844305238343, 0.460727996564}	536.8203916176	0.0015
P16	500	{0.816903951623, 0.164582180338}	6194.3817254403	0.0032
P17	500	{0.478124601239, 0.957209048476}	1019.3526027819	0.0041
P18	500	{0.95138220166, 0.698523640367}	4667.4820432420	0.0036

Table 3: Comparison of the best solutions obtained from the FA, E-FA, ABC, and CB-ABC algorithms for 18 test instances over 30 runs.

Prob	D	FA	E-FA	ABC	CB-ABC
P1	5	38.6308975997	**38.6308975913**	38.6309142364	**38.6308975913**
P2	5	32.6409719983	**32.6409719926**	32.6409728972	**32.6409719926**
P3	5	32.8210027975	**32.8210021279**	**32.8210021279**	**32.8210021279**
P4	10	18.7528484406	**18.7528484372**	18.7528484934	**18.7528484372**
P5	10	18.3972444320	**18.3972444317**	18.3972444324	**18.3972444317**
P6	10	105.9917830548	**105.9917830153**	105.9917835005	**105.9917830153**
P7	50	723.0047799356	723.0047449044	**723.0047448967**	**723.0047448967**
P8	50	108.9894160829	108.9894138822	**108.9894138815**	**108.9894138815**
P9	50	455.1344837962	455.1344639665	**455.1344639631**	**455.1344639631**
P10	100	231.0855604428	231.0855570172	231.0855570166	**231.0855570165**
P11	100	297.2586347001	297.2586211173	297.2586217396	**297.2586211143**
P12	100	286.7386294462	286.7386225632	**286.7386225626**	**286.7386225626**
P13	250	549.9466005355	549.9465893424	549.9476076576	**549.9465893411**
P14	250	543.7610916668	543.7610648272	543.7611199289	**543.7610648263**
P15	250	536.8204201801	536.8203916186	536.8203922596	**536.8203916167**
P16	500	6193.7933019446	6193.7927947779	**6193.7927947450**	**6193.7927947450**
P17	500	1017.8088547714	1017.8088230476	1017.8091304405	**1017.8088230326**
P18	500	4667.0497043757	4667.0492697127	4667.0515567517	**4667.0492696773**

Results of two-sample t-tests are given in Table 5, and they show that the CB-ABC is significantly better than the FA, E-FA, and ABC on 18, 14, and 12 test problems, respectively. It is similar to the FA, E-FA, and ABC on 0, 4, and 6 problems, respectively. It is worth noting that the FA, E-FA, and ABC can not outperform the CB-ABC on any problem. Further, it can be observed that the E-FA is significantly better than the FA on each test problem. In comparison with the ABC, the E-FA is superior on 11 test problems, inferior on 4 problems, and similar on 3 benchmarks. When comparing the performances of the FA and ABC it can be noticed that the FA is significantly better than the ABC on 7 problems, while it is inferior to it on 7 problems. The FA and ABC show similar performances on 4 benchmarks.

According to the results reported in Tables 3, 4, and 5, we can conclude that the CB-ABC and E-FA exhibit superior performances compared to both original versions, ABC and FA, in solving constrained Weber problems with

Table 4: Comparison of the mean values and standard deviations obtained from the FA, E-FA, ABC, and CB-ABC algorithms for 18 test instances over 30 runs.

Prob	D	Stats	FA	E-FA	ABC	CB-ABC
P1	5	Mean	38.6308978682	38.6308975914	38.6310183806	**38.6308975913**
		Std	$2.40E-7$	$4.15E-11$	$1.56E-4$	**2.08E − 14**
P2	5	Mean	32.6409720879	**32.6409719926**	32.6409910473	**32.6409719926**
		Std	$8.97E-8$	$1.57E-11$	$2.80E-5$	**1.21E − 14**
P3	5	Mean	32.8210056512	32.8210021284	**32.8210021279**	**32.8210021279**
		Std	$1.90E-6$	$3.29E-10$	$4.49E-11$	**1.20E − 12**
P4	10	Mean	18.7528484769	**18.7528484372**	18.7528528692	**18.7528484372**
		Std	$1.14E-8$	$6.66E-12$	$5.62E-6$	**1.02E − 14**
P5	10	Mean	18.3972444465	**18.3972444317**	18.3972444546	**18.3972444317**
		Std	$3.40E-8$	$3.77E-12$	$1.91E-8$	**3.04E − 15**
P6	10	Mean	105.9917835910	105.9917830154	105.9917985650	**105.9917830153**
		Std	$4.27E-7$	$9.71E-11$	$1.47E-5$	**1.66E − 14**
P7	50	Mean	723.0049108305	723.0047449232	723.0047449186	**723.0047448968**
		Std	$8.68E-5$	$1.33E-8$	$6.16E-8$	**3.38E − 10**
P8	50	Mean	108.9894309395	108.9894138838	**108.9894138815**	**108.9894138815**
		Std	$8.15E-6$	$1.04E-9$	$3.06E-12$	**2.32E − 12**
P9	50	Mean	455.1346382073	455.1344639826	**455.1344639631**	**455.1344639631**
		Std	$8.83E-5$	$5.60E-8$	**1.49E − 13**	$8.79E-13$
P10	100	Mean	231.0855873314	231.0855570223	231.0855570526	**231.0855570166**
		Std	$1.36E-5$	$2.94E-9$	$1.07E-8$	**1.30E − 11**
P11	100	Mean	297.2586695071	297.2586211205	297.2587315275	**297.2586211143**
		Std	$2.23E-5$	$2.39E-9$	$3.19E-4$	**1.03E − 11**
P12	100	Mean	286.7386559609	286.73862256805	**286.7386225626**	**286.7386225626**
		Std	$2.03E-5$	$3.15E-9$	$1.02E-11$	**1.10E − 12**
P13	250	Mean	549.9466493305	549.9465893483	549.9913513828	**549.94658934110**
		Std	$2.40E-5$	$3.27E-9$	0.0527	**3.91E − 11**
P14	250	Mean	543.7611473895	543.7610648386	543.7655305827	**543.7610648263**
		Std	$3.45E-5$	$5.68E-9$	0.00540	**2.51E − 12**
P15	250	Mean	536.8205045637	536.8203916312	536.8204579818	**536.8203916167**
		Std	$7.09E-5$	$8.00E-9$	$1.08E-4$	**2.83E − 12**
P16	500	Mean	6193.7944120365	6193.7927949375	6193.7927947582	**6193.7927947450**
		Std	$7.68E-4$	$1.13E-7$	$2.26E-8$	**1.42E − 11**
P17	500	Mean	1017.8090988841	1017.8088230685	1017.8110427370	**1017.8088230326**
		Std	$1.63E-4$	$1.55E-8$	0.0043	**2.35E − 11**
P18	500	Mean	4667.0511842616	4667.0492697730	4667.0890166945	**4667.0492696773**
		Std	$9.17E-4$	$4.63E-8$	0.0588	**1.33E − 11**

the connected feasible region bounded by arcs. Further, from these results and according to the results from Table 2, it is clear that the CB-ABC outperformed all other three metaheuristic algorithms as well as the heuristic algorithm with respect to the quality of the obtained results. Although the CB-ABC has more accurate and more stable results than the remaining three metaheuristics, all four metaheuristic approaches perform better than or equal to the heuristic approach with respect to the quality of the obtained results for most of the tested problems.

4.4. Computational Time Analysis. In order to compare the computational cost of the four metaheuristic algorithms, we computed the mean of the CPU times over 30 runs taken by each metaheuristic algorithm. These results are reported in Table 6. The results from Table 6 show that the execution time for each of the metaheuristics approaches linearly increases when the number of the constraints or input points increases.

By comparing computational times for the ABC and CB-ABC algorithms with respect to the FA and E-FA, it is observable that ABC and CB-ABC algorithms are about 4 times faster than the FA and about 20 times faster than the E-FA for the majority of test problems. The computational times of the ABC and CB-ABC algorithms are not significantly different. In addition, when the number of constraints is 500 that time is less than 0.1 seconds. The computational time requirements for the E-FA algorithm are about five times greater compared to the FA and when the number

Table 5: The results of 95%-confidence two-sample t-test over each test problem.

Prb	FA versus E-FA		FA versus ABC		FA versus CB-ABC		E-FA versus ABC		E-FA versus CB-ABC		ABC versus CB-ABC	
	FA	E-FA	FA	ABC	FA	CB-ABC	E-FA	ABC	E-FA	CB-ABC	ABC	CB-ABC
P1	−	+	+	−	−	+	+	−	−	+	−	+
P2	−	+	+	−	−	+	+	−	+	+	−	+
P3	−	+	−	+	−	+	−	+	−	+	+	+
P4	−	+	+	−	−	+	+	−	+	+	−	+
P5	−	+	+	+	−	+	+	−	+	+	−	+
P6	−	+	+	−	−	+	+	−	−	+	−	+
P7	−	+	−	+	−	+	+	+	−	+	+	+
P8	−	+	−	+	−	+	−	+	−	+	+	+
P9	−	+	−	+	−	+	+	+	+	+	+	+
P10	−	+	−	+	−	+	+	−	−	+	−	+
P11	−	+	+	+	−	+	+	+	−	+	+	+
P12	−	+	+	+	−	+	−	+	−	+	+	+
P13	−	+	−	+	−	+	+	−	−	+	−	+
P14	−	+	+	−	−	+	+	−	−	+	−	+
P15	−	+	+	+	−	+	+	−	−	+	−	+
P16	−	+	−	+	−	+	−	+	−	+	−	+
P17	−	+	+	−	−	+	+	−	−	+	−	+
P18	−	+	+	−	−	+	+	−	−	+	−	+
Total	0	18	11	11	0	18	14	7	4	18	6	18

Table 6: Mean of the CPU times (in seconds) obtained from the FA, E-FA, ABC, and CB-ABC algorithms for 18 test instances over 30 runs.

Prob	D	FA	E-FA	ABC	CB-ABC
P1	5	0.022	0.036	0.004	0.004
P2	5	0.020	0.044	0.005	0.004
P3	5	0.022	0.040	0.004	0.004
P4	10	0.025	0.060	0.005	0.006
P5	10	0.022	0.060	0.006	0.006
P6	10	0.025	0.062	0.004	0.005
P7	50	0.056	0.226	0.013	0.016
P8	50	0.052	0.224	0.014	0.014
P9	50	0.057	0.252	0.014	0.010
P10	100	0.090	0.690	0.035	0.043
P11	100	0.090	0.738	0.034	0.043
P12	100	0.086	0.544	0.022	0.025
P13	250	0.200	0.990	0.044	0.044
P14	250	0.216	1.316	0.048	0.051
P15	250	0.194	1.046	0.045	0.042
P16	500	0.376	2.182	0.082	0.093
P17	500	0.360	1.990	0.080	0.091
P18	500	0.374	2.128	0.081	0.094

of constraints or input points is 500 that time is about two seconds.

Compared with the computational time results of the heuristic approach, which are presented in Table 2, it can be seen that the heuristic algorithm requires less computational time than the four metaheuristic algorithms. However, the computational time of the four metaheuristics is reasonable and it can be considered as negligible, since it is less than one second in most cases.

5. Conclusion

The constrained Weber problem with feasible region bounded by arcs represents a problem of a nonconvex optimization.

Finding a global optimum of such a problem is difficult considering the fact that it has multiple locally optimal points within the feasible region. Metaheuristic approaches for solving this problem are suitable choice, since these techniques can obtain quality results in a reasonable amount of time.

The performances of two prominent swarm-intelligence algorithms (the artificial bee colony and firefly algorithm) and their recently proposed improved versions for constrained optimization (the crossover-based artificial bee colony and enhanced firefly algorithm) are compared. The heuristic algorithm based on modified Weiszfeld procedure is also implemented for the purpose of the comparison with the metaheuristic approaches.

The four metaheuristic algorithms are compared on eighteen randomly generated test instances in which the number of input points or constraints increases up to 500. Numerical results indicate that all four metaheuristic algorithms are superior compared to the heuristic approach with respect to the precision of the results, with the notable ascendancy of the CB-ABC algorithm. In terms of the execution time, the ABC and CB-ABC are more efficient than the FA and E-FA. Although these four algorithms require somewhat higher computational cost than the heuristic approach, the CPU times for all these algorithms are reasonable and grow at a linear rate as the number of input points or constraints increases. Finally, it turns out that the CB-ABC algorithm is superior compared to other metaheuristics with respect to the quality of the results, robustness, and computational efficiency.

From this research it can be concluded that metaheuristic approaches can be successfully used for problems with maximum and minimum distance limits. Further, this research encourages the application of the metaheuristic algorithms for solving some other complex constrained optimization problems of practical importance.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The second and third authors gratefully acknowledge support from the Project supported by Ministry of Education and Science of Republic of Serbia, Grant no. 174013. The first, third, and fifth authors gratefully acknowledge support from the Project Applying Direct Methods for Digital Image Restoring of the Goce Delčev University. The fourth author gratefully acknowledges support from the Ministry of Education and Science of Russian Federation (Project 2.5527.2017/8.9).

References

[1] R. Z. Farahani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms and Case Studies*, Springer-Verlag, Berlin, Germany, 2009.

[2] A. Weber, *Ueber den Standort der Industrien, Erster Teil: Reine Theorie des Standortes*, J.C.B. Mohr, Tübingen, Germany, 1909.

[3] G. Wesolowsky, "The Weber problem: history and perspectives," *Location Science*, vol. 1, pp. 5–23, 1993.

[4] Z. Drezner, K. Klamroth, A. Schöbel, and G. O. Wesolowsky, "The Weber problem," in *Facility Location*, pp. 1–36, Springer, Berlin, Germany, 2002.

[5] P. S. Stanimirović, M. S. Ćirić, L. A. Kazakovtsev, and I. A. Osinuga, "Single-facility Weber location problem based on the lift metric," *Facta Universitatis. Series: Mathematics and Informatics*, vol. 27, no. 2, pp. 175–190, 2012.

[6] R. F. Love, W. G. Truscott, and J. Walker, "Facilities Location: Models and Methods," *International Journal of Machine Learning and Cybernetics, North-Holland, New York*, 1988.

[7] E. Weiszfeld, "Sur le point sur lequel la somme des distances de n points donnes est minimum," *Tohoku Mathematical Journal*, vol. 43, no. 1, pp. 335–386, 1937.

[8] Y. Vardi and C.-H. Zhang, "A modified Weiszfeld algorithm for the Fermat-WEBer location problem," *Mathematical Programming*, vol. 90, no. 3, Ser. A, pp. 559–566, 2001.

[9] C. Szegedy, *Some Applications of the Combinatorial Laplacian*, University of Bonn, 2005.

[10] P. Hansen, N. Mladenović, and E. Taillard, "Heuristic solution of the multisource Weber problem as a p-median problem," *Operations Research Letters*, vol. 22, no. 2-3, pp. 55–62, 1998.

[11] S. Gonzalez-Martin, A. Ferrer, A. A. Juan, and D. Riera, "Solving non-smooth arc routing problems throughout biased-randomized heuristics," *Advances in Intelligent Systems and Computing*, vol. 262, pp. 451–462, 2014.

[12] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic algorithms in modeling and optimization," in *Metaheuristic Applications in Structures and Infrastructures*, A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, Eds., pp. 1–24, Elsevier, 2013.

[13] R. Martí and G. Reinelt, *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*, vol. 175, Springer, New York, NY, USA, 2011.

[14] A. Afshar, F. Massoumi, A. Afshar, and M. A. Mariño, "State of the art review of ant colony optimization applications in water resource management," *Water Resources Management*, vol. 29, no. 11, pp. 3891–3904, 2015.

[15] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. I. Background and development," *Natural Computing. An International Journal*, vol. 6, no. 4, pp. 467–484, 2007.

[16] I. Fister Jr., M. Perc, S. M. Kamal, and I. Fister, "A review of chaos-based firefly algorithms: perspectives and research challenges," *Applied Mathematics and Computation*, vol. 252, pp. 155–165, 2015.

[17] I. Fister, X. S. Yang, and D. Fister, "Cuckoo search: a brief literature review," in *Cuckoo Search and Firefly Algorithm: Theory and Applications*, Xin-She Yang, Ed., pp. 49–62, Springer International Publishing.

[18] I. Fister, I. Fister Jr., X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, no. 1, pp. 34–46, 2013.

[19] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[20] M. Bilal, M. Shams-ur-Rehman, and M. A. Jaffar, "Reconstruction: Image restoration for space variant degradation," *Smart Computing Review*, vol. 3, no. 4, pp. 220–232, 2013.

[21] P. Siarry, *Optimisation in Signal and Image Processing*, Wiley-ISTE, 2009.

[22] X. Yang, S. F. Chien, and T. O. Ting, "Computational Intelligence and Metaheuristic Algorithms with Applications," *The Scientific World Journal*, vol. 2014, Article ID 425853, 4 pages, 2014.

[23] I. Fister Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *Elektrotehniški Vestnik*, vol. 80, no. 3, pp. 1–7, 2013.

[24] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[25] B. C. Mohan and R. Baskaran, "A survey: ant colony optimization based recent research and implementation on several engineering domain," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4618–4627, 2012.

[26] R. F. Tavares Neto and M. Godinho Filho, "Literature review regarding ant colony optimization applied to scheduling problems: guidelines for implementation and directions for future research," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 150–161, 2013.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[28] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. II. Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing. An International Journal*, vol. 7, no. 1, pp. 109–124, 2008.

[29] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in Solar Photovoltaic system," *Journal Applied Soft Computing*, vol. 13, no. 5, pp. 2997–3006, 2013.

[30] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[31] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[32] A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem," in *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, F. T. S. Chan and M. K. Tiwari, Eds., pp. 113–144, I-Tech Education and Publishing, Vienna, Austria, 2007.

[33] S. Zhang and S. Liu, "A novel artificial bee colony algorithm for function optimization," *Mathematical Problems in Engineering*, vol. 2015, Article ID 129271, 10 pages, 2015.

[34] Y. Liang, Z. Wan, and D. Fang, "An improved artificial bee colony algorithm for solving constrained optimization problems," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 3, pp. 739–754, 2017.

[35] T. K. Sharma and M. Pant, "Shuffled Artificial Bee Colony Algorithm," *Soft Computing*, pp. 1–20, 2016.

[36] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, 2009.

[37] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Computers and Structures*, vol. 89, no. 23-24, pp. 2325–2336, 2011.

[38] M. Huang, J. Yuan, and J. Xiao, "An adapted firefly algorithm for product development project scheduling with fuzzy activity duration," *Mathematical Problems in Engineering*, vol. 2015, Article ID 973291, 11 pages, 2015.

[39] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[40] Y. Lin, C. Zhang, and Z. Liang, "Cuckoo search algorithm with hybrid factor using dimensional distance," *Mathematical Problems in Engineering*, vol. 2016, Article ID 4839763, 11 pages, 2016.

[41] A. R. Yildiz, "Cuckoo search algorithm for the selection of optimal machining parameters in milling operations," *International Journal of Advanced Manufacturing Technology*, vol. 64, no. 1–4, pp. 55–61, 2013.

[42] M. Hakan Akyüz, T. Öncan, and I. Kuban Altinel, "Beam search heuristics for the single and multi-commodity capacitated multi-facility weber problems," *Computers and Operations Research*, vol. 40, no. 12, pp. 3056–3068, 2013.

[43] J.-L. Jiang and X.-M. Yuan, "A heuristic algorithm for constrained multi-source Weber problem—the variational inequality approach," *European Journal of Operational Research*, vol. 187, no. 2, pp. 357–370, 2008.

[44] L. A. Kazakovtsev, "Algorithm for constrained Weber problem with feasible region bounded by arcs," *Facta Universitatis. Series: Mathematics and Informatics*, vol. 28, no. 3, pp. 271–284, 2013.

[45] M. Luis, S. Salhi, and G. Nagy, "Region-rejection based heuristics for the capacitated multi-source Weber problem," *Computers and Operations Research*, vol. 36, no. 6, pp. 2007–2017, 2009.

[46] S. M. H. Manzour-Al-Ajdad, S. A. Torabi, and K. Eshghi, "Single-source capacitated multi-facility weber problem—an iterative two phase heuristic algorithm," *Computers and Operations Research*, vol. 39, no. 7, pp. 1465–1476, 2012.

[47] M. Bischoff and K. Klamroth, "An efficient solution method for Weber problems with barriers based on genetic algorithms," *European Journal of Operational Research*, vol. 177, no. 1, pp. 22–41, 2007.

[48] A. Ghaderi, M. S. Jabalameli, F. Barzinpour, and R. Rahmaniani, "An efficient hybrid particle swarm optimization algorithm for solving the uncapacitated continuous location-allocation problem," *Networks and Spatial Economics*, vol. 12, no. 3, pp. 421–439, 2012.

[49] H. G. Gharravi and M. S. Farham, "Applying metaheuristic approaches on the single facility location problem with polygonal barriers," *International Journal of Metaheuristics*, vol. 3, no. 4, p. 348, 2014.

[50] M. Saleh Farham, H. Süral, and C. Iyigun, "The Weber problem in congested regions with entry and exit points," *Computers and Operations Research*, vol. 62, pp. 177–183, 2015.

[51] N. Javadian, R. Tavakkoli-Moghaddam, M. Amiri-Aref, and S. Shiripour, "Two meta-heuristics for a multi-period minisum location-relocation problem with line restriction," *International Journal of Advanced Manufacturing Technology*, vol. 71, no. 5-8, pp. 1033–1048, 2014.

[52] N. Mohammadi, M. R. Malek, and A. A. Alesheikh, "A new GA based solution for capacitated multi source Weber problem," *International Journal of Computational Intelligence Systems*, vol. 3, no. 5, pp. 514–521, 2010.

[53] D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing Journal*, vol. 11, no. 3, pp. 3021–3031, 2011.

[54] I. Brajevic, "Crossover-based artificial bee colony algorithm for constrained optimization problems," *Neural Computing and Applications*, vol. 26, no. 7, pp. 1587–1601, 2015.

[55] I. Brajevic and J. Ignjatović, "An enhanced firefly algorithm for mixed variable structural optimization problems," *Facta Universitatis. Series: Mathematics and Informatics*, vol. 30, no. 4, pp. 401–418, 2015.

[56] L. A. Kazakovtsev and P. S. Stanimirovic, "Algorithm for Weber problem with a metric based on the initial fare," *Journal of Applied Mathematics & Informatics*, vol. 33, no. 1-2, pp. 157–172, 2015.

[57] Z. Drezner, C. Scott, and J.-S. Song, "The central warehouse location problem revisited," *IMA Journal of Management Mathematics*, vol. 14, no. 4, pp. 321–336 (2004), 2003.

[58] P. Hansen, D. Peeters, and J. F. Thisse, "Algorithm for a constrained weber problem," *Management Science*, vol. 28, no. 11, pp. 1285–1295, 1982.

[59] P. Hansen, D. Peeters, and J.-F. Thisse, "Constrained Location and the Weber-Rawls Problem," *North-Holland Mathematics Studies*, vol. 59, no. C, pp. 147–166, 1981.

[60] D. Karaboga and B. Basturk, *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*, Springer-Verlag, Berlin, Germany, LNAI 4529: IFSA'07, 2007.

[61] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[62] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2010.

[63] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu, "A framework for self-tuning optimization algorithm," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2051–2057, 2013.

[64] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[65] M. Mernik, S.-H. Liu, D. Karaboga, and M. Crepinšek, "On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation," *Information Sciences*, vol. 291, pp. 115–127, 2015.

[66] I. Fister, X. Yang, J. Brest, and I. Fister Jr., "Modified firefly algorithm using quaternion representation," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7220–7230, 2013.

[67] E. Mezura-Montes and O. Cetina-Domínguez, "Empirical analysis of a modified artificial bee colony for constrained numerical optimization," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 10943–10973, 2012.